Report on the CNN Code
PD-Internship
Lerner Paul
06/06/2018

Inspired by the works of Oh et al. and Bai et al. (cf. Report #3 and #4 respectively), I tried to use 1d CNN instead of a RNN.
Our work is then similar to Pereira et al. as we interpret the sensors signals as an image but instead of resizing it to a 2d, square image we keep it 1d. Also, as them, we perform deep, end-to-end learning (we're the only two to do so on PD diagnosis via HaW, to the best of my knowledge, cf. report #2).

Overview

1 Model architecture **1**

2 Model training **3**

2.1 Batch training 3

3 Visualization & Interpretation **4**

3.1 Convolutional Layer 4

Conclusion - Todo List **8**

1 Model architecture

Hypothesis :

1. As a 2d conv takes the spatial aspect of the data into account, a 1d conv takes the sequential aspect of the data into account
2. We don't need to predict a sequence so we don't need to pad the CNN layers or use causal convolutions as Bai et al. (cf. Report #4)
3. dilated convolutions allow the model to have a "large scope" before taking a decision (see Bai et al.).
4. a deep network will a priori extract advanced/complex features and we have don't have enough data for that.
5. we should aim for a feature map between 5 and 200 neurons, it's the number of features used by Mucha et al. and Drotar et al., respectively (cf. report #2)
6. stride is only good to reduce the dimensionality of the data/features

7. using a strided convolution or a strided pooling seems to be equivalent, therefore, intuitively, I prefer to use strided pooling and keep the stride of conv layer(s) at 1.

Questions :

1. how large should the kernel(s) be ?
2. should we first have a high dimensional data (i.e. with numerous filters) that we then pool or start with a small number of filters ?
3. how dilated should the conv(s) be ? Do the network needs to have a large scope ? I will first experiment without dilation.

In order to have the same sequence length on all samples I padded and trimmed the samples to 3117 timesteps (3rd quartile). We could also keep the data untrimmed and unpadded then pool with a kernel of the same size as the sequence length dimension in order to have only one neuron per filter. That's what I do in the CNN-baseline, see below :

CNN100-baseline :

- only 1 conv layer with 100 kernels of size 1 (thus stride and dilation of 1) and ReLU activation and weight_norm regularization
- one pooling layer with a filter of 16071 (max length of the spiral) thus reducing the sequence to 1.
- dropout with probability 20%
- one FC layer (fed the feature map of size 100) with a sigmoid activation

Later I might refer to this model as CNN***-baseline with *** being the number of kernels.

In order to first reduce the dimensionality of the data, I downsampled it 10 times. This had bad effect on the CNN-baseline.

Strangely, I achieve better results when using a single layer with a kernel size of 1 (thus not taking into account the sequential aspect of the data) then any other configurations (with less or equal n° of total parameters). This might be explained in 2 ways :

1. We don't need to take the sequential aspect of the data
2. We don't have enough data to learn the the sequential features thus we must keep on using static features.

As with RNN, I tried to build a hierarchical CNN on the *l* task : I split the task into letters (i.e. *l*s). See figure below. Unlike the previous RNN experiment, here I individually standardized each letter. Moreover, I padded each letter to 1242 timesteps (max. length). Since the subjects n° 12, 21, 23, 44, 67 did 6 *l* instead of 5 I discarded their last *l*.

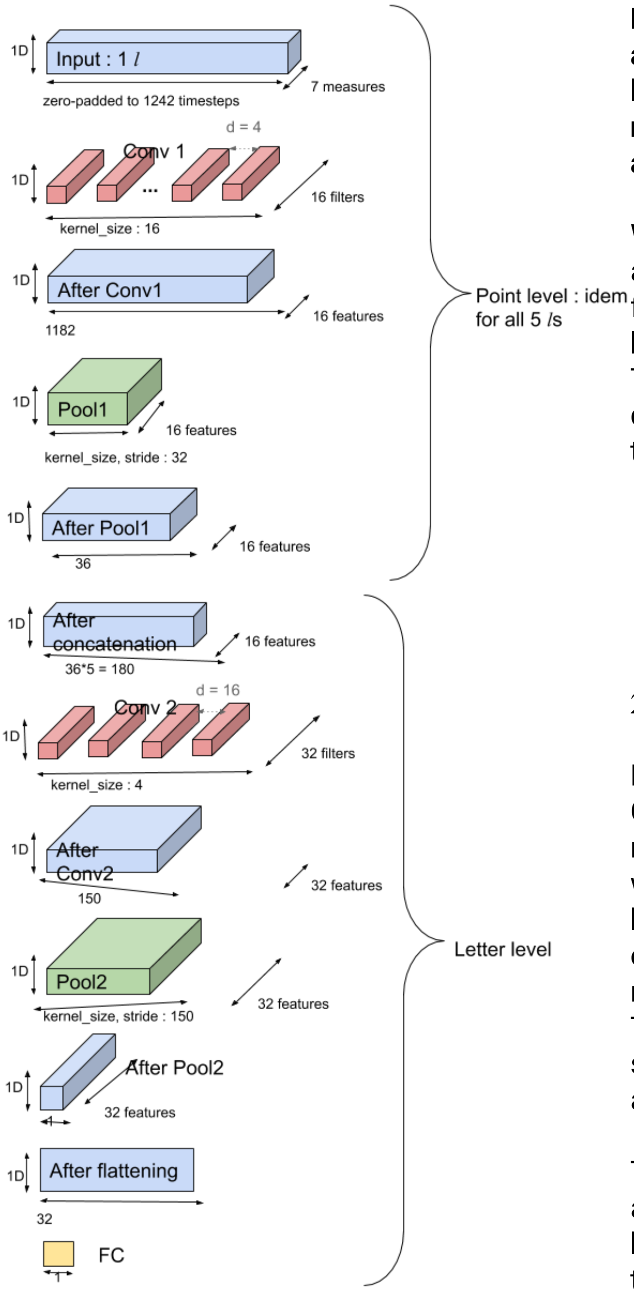


Fig. Hierarchical CNN. A dropout of 0.1 is applied after the pooling layers. Both Conv layers have ReLU activation and weight norm regularization. FC layer has Sigmoid activation.

With this Hierarchical CNN we're able to achieve 65% accuracy on 10-CV. This is the first time we're able to go above chance level with the *l* task.
The hyperparameters were found with a quick random search (~10 experiments) in the following ranges :

- learning_rate : {0.001, 0.0001}
- dropout : {0, 0.1, 0.2, 0.3, 0.4, 0.5}
- hidden_size : {4, 8, 16, 32}
- dilation : {1, 2, 4, 8, 16}
- conv_kernel : {1, 2, 4, 8, 16}
- pool_kernel : {1, 2, 4, 8, 16, 32, 64, 128, 256, 512}

I compared this model with an equivalent CNN which was fed the whole *l* task. The model is the same except that the input is whole *l* task padded to 4226 timesteps (max length) so there's no concatenation obviously. Moreover, in order to provide roughly the same number of parameters, The Pool1 has a stride of 24 instead of 32 so the output length would be 173, as close as possible to 180.

This equivalent model achieves 58% accuracy on 10 CV so it suggests that the letter split augments the dataset and allows the hierarchical model to generalize better than the simple model.

2 Model training

2.1 Batch training

With CNNs, we need the input data to be equally long (cf. 1 Model architecture). This allows for a batch training, thus I trained the model on batches of the size of the training and test set respectively, regardless of the number of subjects in it. It divided the training time by 5 on the CNN50-baseline. However, as I've read [add ref], the model is less efficient as the loss is averaged thus less relevant. See figures below.

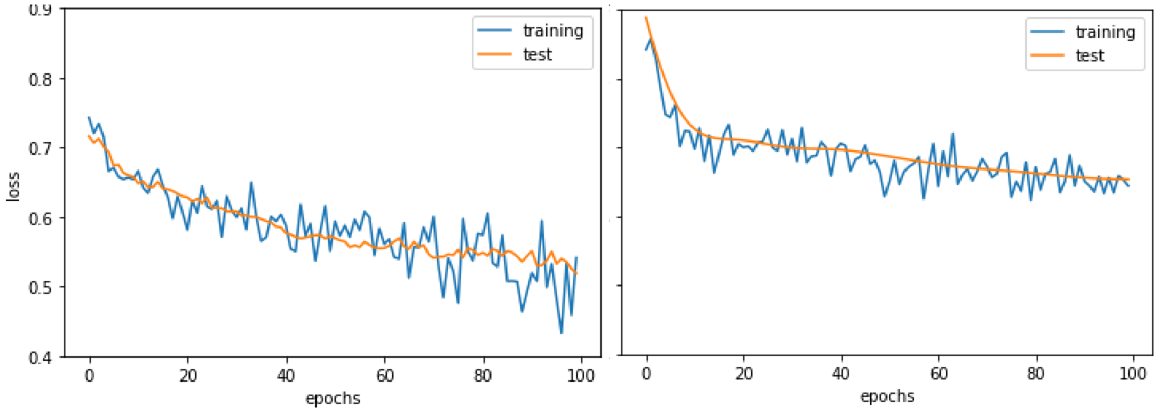
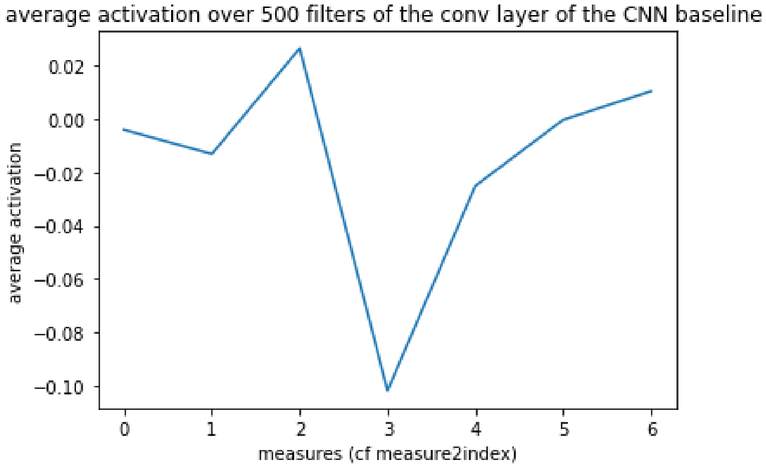


Fig. CNN50-baseline, left : training on a single subject, right : batch training.

3 Visualization & Interpretation

3.1 Convolutional Layer



With the CNN500-baseline, after training on the last fold for 100 epochs (achieving 83% accuracy on the last epoch) we can see that, in average, the neurons corresponding to the button_status measures are less activated than other. This is probably because most of button status are at 1 (i.e. on paper) for the spiral task we trained on, thus there's no need for 500 filters to monitor it. Moreover, this measure is redundant with pressure. This could also be explained by the binary nature of the variable, it contains less information then other measures. The shape of the plot is similar with the hCNN on the *l* task which suggests that my first hypothesis is wrong.

As the classification accuracy suggests, 500 seems to be a too big number of filters, several of them seems to be monitoring the same thing :

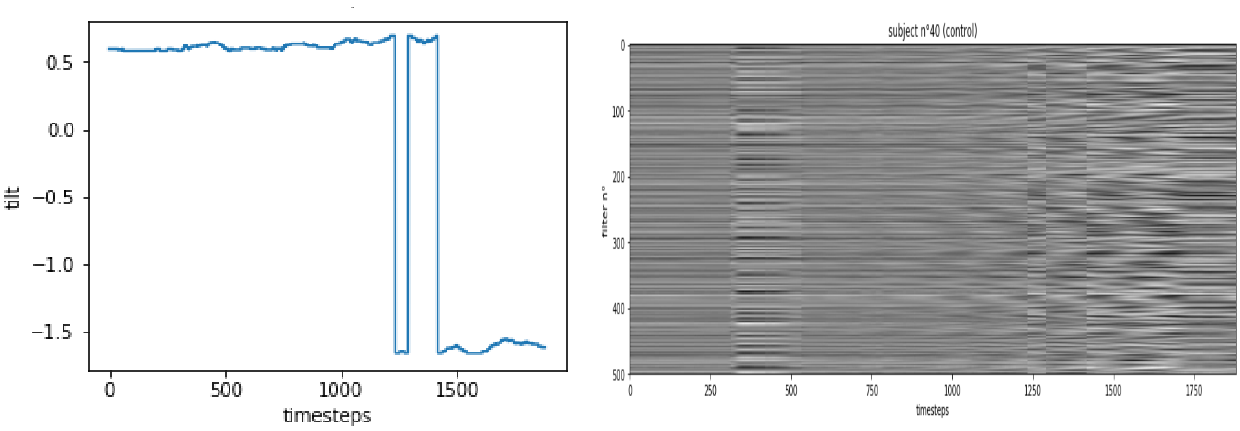


Fig. Subject n°40 (Control) Left : Tilt vs time, right : after a forward pass in the convolution layer (white means high activation, black low).

As you can see in the figure above, a lot of neurons seems to be highly activated when there's a drop in the tilt, see below for a zoom.

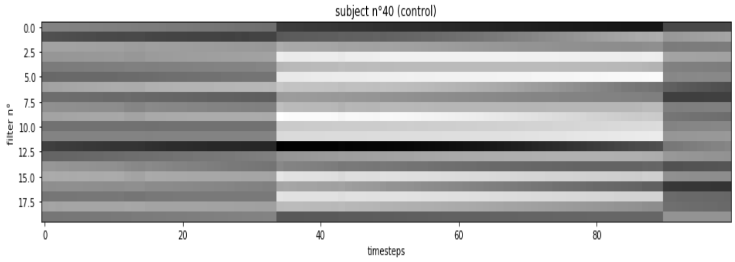


Fig. Zoom of the above figure : feature activation on filters 400 to 420 and timesteps 1200 to 1300.

As you can see, several filters activate at the exact same time : when there's a drop in the tilt measure (this has also been observed with subject n°8 (PD)). See below for details of filters 403 and 405.

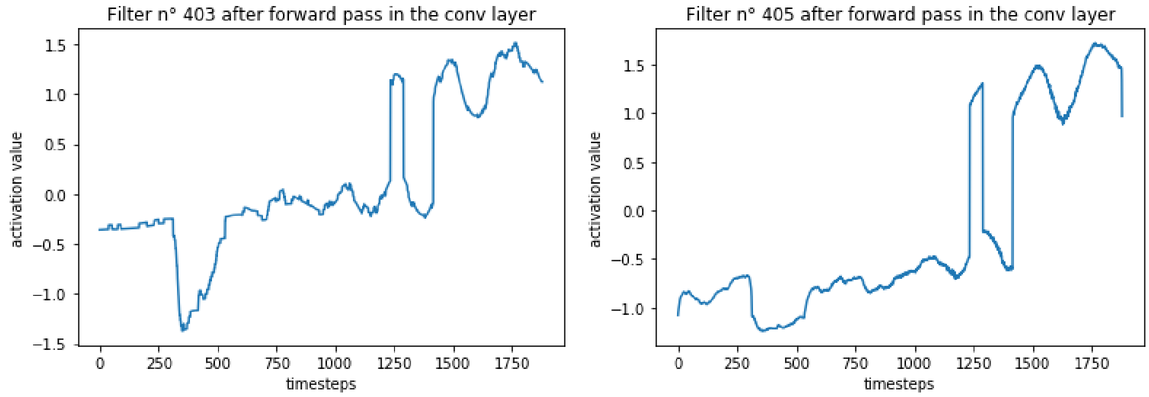


Fig. Activation of filters 403 and 405 (left and right, respectively).

Similar observations wan be made when there's drop in other measure's, e.g. pressure or button_status (see average activation plot). Some filters seems to monitor different measure at the same time, see below.

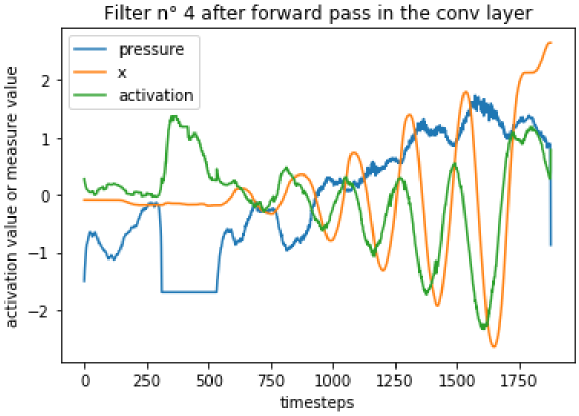


Fig. Filter 4 plotted along pressure and x-coordinate. We can see that the filter first activates when there's a drop in the pressure but then seems to follow the x coordinate.

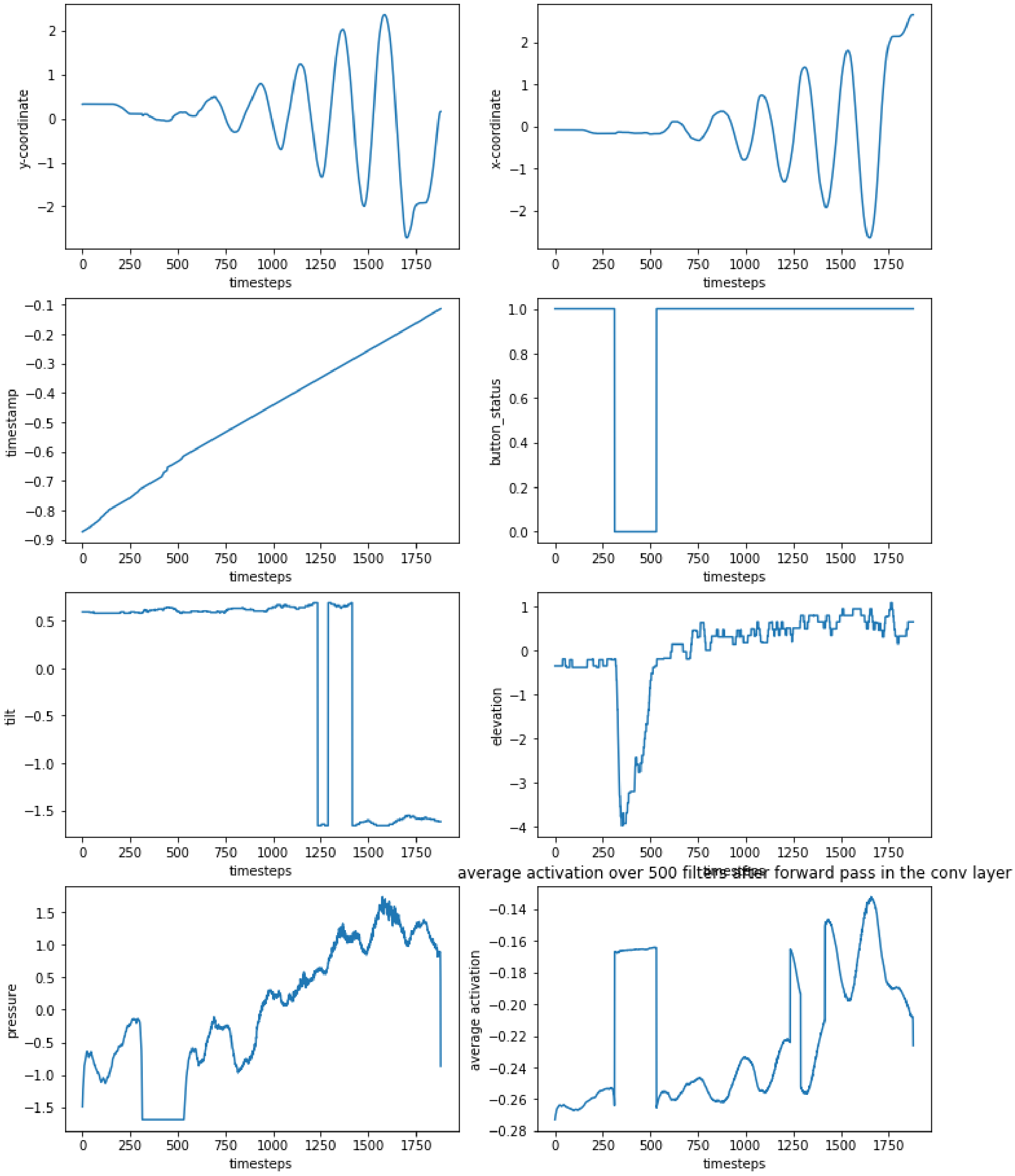


Fig. Subject's 40 (PD) measures vs time and average activation after forward pass in the conv layer (bottom right). We can see extremums in the average activation that match extremums in the measures (e.g peak around 1250 timesteps that we have observed above).

The first conv layer of the hCNN provides similar insights but the second layer is more difficult to interpret as it's more abstract.

Conclusion - Todo List

These first experiments with CNNs provide encouraging results : we now outperform Moetesum et al. on the l task and Drotar et al. on the spiral task !

I think I should probably discuss and compare my results with Catherine. It should be interesting to compare her model (CNN2d that takes as input Gramian angular field or spectrogram) and mine. By the way I wonder if she knows the work of Afonso et al. and why motivated her use of Short Time Fourier Transform over Discrete wavelet transform.

I also wanted to extract some features from the data, e.g. speed.